AMENDMENTS TO THE CLAIMS

The following listing of claims will replace all prior versions and listings of claims in the Application:

Listing of Claims:

1. (currently amended) A method comprising:

determining a set of arguments for an outsourced computation;

classifying, with a first computer, ~~the~~ said outsourced computation into at least one ~~of a number of computation types~~ computation type, said at least one computation type being selected from the group consisting of quadrature computations, image edge detection computations, convolution computations, character string pattern matching computations, sorting computations, and computations for solving one or more differential equations;

selecting, with ~~the~~ said first computer, one or more disguising operations from a predetermined set of disguising operations based on said classifying;

performing ~~the~~ said one or more selected disguising operations on ~~the~~ said actual arguments with ~~the~~ said first computer to provide disguised arguments;

outputting ~~the~~ said disguised arguments from ~~the~~ said first computer for performance of ~~the~~ said outsourced computation; and

receiving a result of ~~the~~ said outsourced computation performed with ~~the~~ said disguised arguments.

2. (currently amended) The method of claim 1, further comprising computing an actual answer from ~~the~~ said result after said receiving.

3. (cancelled)

4. (cancelled)

5. (cancelled)

6. (currently amended) The method of claim 1, further comprising:

receiving ~~the~~ said disguised arguments at a second computer remotely located relative to ~~the~~ said first computer;

performing ~~the~~ said outsourced computation with ~~the~~ said second computer; and

sending ~~the~~ said result from ~~the~~ said second computer to ~~the~~ said first computer, ~~the~~ said result being in a disguised form relative to an answer obtained by submitting ~~the~~ said actual arguments to ~~the~~ said outsourced computation.

7. (currently amended) The method of claim 1, wherein said ~~preparing includes~~ step of performing said one or more selected disguising operations comprises the step of generating a plurality of ~~random~~ pseudorandom numbers, ~~the random~~ each of said plurality of pseudorandom numbers ~~each~~ being generated by one of a number of ~~random~~ pseudorandom number generation techniques, ~~the~~ said techniques each comprising a different distribution parameter.

8. (currently amended) The method of claim 7, wherein said ~~preparing further~~ step of performing said one or more selected disguising operations comprises defining a number of disguise functions with one or more of ~~the random~~ said pseudorandom numbers.

9. (currently amended) The method of claim 1, wherein said ~~preparing~~ step of performing said one or more selected disguising operations comprises modifying a linear operator.

10. (currently amended) The method of claim 1, wherein said ~~preparing~~ step of performing said one or more selected disguising operations comprises altering a dimension corresponding to ~~the~~ said actual arguments to provide ~~the~~ said disguised arguments.

11. (currently amended) The method of claim 10, wherein said altering comprises expanding ~~the~~ said dimension.

12. (currently amended) The method of claim 1, wherein said ~~preparing~~ step of performing said one or more selected disguising operations comprises performing a function substitution in accordance with at least one mathematical identity.

13. (cancelled)

14. (cancelled)

15. (cancelled)

16. (cancelled)

17. (cancelled)

18. (currently amended) A system comprising:

a computer operable to define a set of actual arguments for an outsourced computation, said computer being programmed to classify said computation into at least one ~~of a plurality of computation types~~ computation type, said at least one computation type being selected from the group consisting of quadrature computations, image edge detection computations, convolution

computations, character string pattern matching computations, sorting computations, and computations for solving one or more differential equations, said computer being programmed to determine a group of disguised arguments from ~~the~~ said set of actual arguments, said disguised arguments hiding one or more characteristics of ~~the~~ said set of actual arguments;

an output device responsive to said computer to output ~~the~~ said disguised arguments for remote performance of said outsourced computation; and

an input device to receive a result of said outsourced computation performed with said disguised arguments, wherein said computer is responsive to said input device to determine a desired answer from said result.

19. (cancelled)

20. (original) The system of claim 18, further comprising a computing center, said computing center being programmed to perform said outsourced computation with said disguised arguments.

21. (original) The system of claim 18, wherein said computer includes a memory, a library of disguise operations being stored in said memory, said computer programming referencing said library to generate said disguised arguments.

22. (original) The system of claim 21, wherein said disguise operations correspond to at least one of the group consisting of random object generation, argument dimension modification, mathematical identity substitution, and disguise function generation.

23. (original) The system of claim 18, wherein said computer includes instructions to generate a cubic spline to provide a disguise for said actual arguments.

24. (cancelled)

25. (cancelled)

26. (cancelled)

27. (cancelled)

28. (currently amended) An apparatus, comprising: a computer readable medium, said medium defining computer programming instructions to hide a group of actual arguments for a computation to be outsourced, said programming instructions being operable to classify said computation into at least one ~~of a plurality of computation types, said~~ computation type, said at least one computation type being selected from the group consisting of quadrature computations, image edge detection computations, convolution computations, character string pattern matching computations, sorting computations, and computations for solving one or more differential equations, said programming instructions being operable to generate a group of disguised arguments corresponding to said actual arguments, said disguised arguments being generated to ~~provide~~ produce a disguised result ~~when provided for~~ from said computation, an actual answer being recoverable from said disguised result in accordance with said programming instructions, said actual answer being returned by said computation when said computation is provided said actual arguments.

29. (currently amended) The apparatus of claim 28, further ~~including~~ comprising a computer responsive to said programming instructions.

30. (cancelled)

31. (cancelled)

32. (original) The apparatus of claim 28, wherein said programming instructions define a routine to generate a cubic spline to provide at least one disguise function.

33. (currently amended) The apparatus of claim 28, wherein said programming instructions define a routine to provide a ~~random~~ pseudorandom function space to provide one or more disguise functions.

34. (previously presented) A system comprising:

a computer operable to define a set of actual arguments for an outsourced computation, said computer being programmed to determine a group of disguised arguments from said set of actual arguments, said computer comprising instructions to generate a cubic spline to provide a

disguise for said actual arguments, said disguised arguments hiding one or more characteristics of said set of actual arguments;

an output device responsive to said computer to output said disguised arguments for remote performance of said outsourced computation;

an input device to receive a result of said outsourced computation performed with said disguised arguments; and

wherein said computer is responsive to said input device to determine a desired answer from said result.

35. (previously presented) An apparatus comprising:

a computer readable medium, said medium comprising computer programming instructions to hide a group of actual arguments for a computation to be outsourced, said programming instructions being operable to generate a group of disguised arguments corresponding to said actual arguments, said programming instructions comprising a routine to generate a cubic spline to provide at least one disguise function, said disguised arguments being generated to provide a disguised result when provided for said computation, an actual answer being recoverable from said disguised result in accordance with said instructions, said actual answer being returned by said computation when said computation is provided said actual arguments.

36. (previously presented) The method of claim 1, wherein said preparing comprises performing a function substitution in accordance with at least one expansion of unity.

37. (currently amended) A method for outsourcing a matrix multiplication computation from a first computer to a second computer, wherein the contents of a matrix are disguised prior to delivery of the matrix to the second computer for the matrix multiplication computation thereby hindering discovery of the contents of the matrix by the second computer and unauthorized parties, the method comprising the steps of:

providing, in a memory of a first computer, a first actual matrix M1 comprising a first plurality of actual arguments, and a second actual matrix M2 comprising a second plurality of actual arguments, wherein a multiplicative product of said first actual matrix M1 and said second actual matrix M2 is desired;

preparing, in said memory of said first computer, at least two disguising matrices, each said disguising matrix being a sparse matrix comprising at least one non-zero disguising argument, wherein each said at least one non-zero disguising argument comprises a ~~random~~ pseudorandom number;

disguising said first plurality of actual arguments by computing, with said first computer, a first disguised matrix X using said first actual matrix M1 and at least one said disguising matrix, said first disguised matrix X comprising a first plurality of disguised arguments; and

disguising said second plurality of actual arguments by computing, with said first computer, a second disguised matrix Y using said second actual matrix M2 and at least one said disguising matrix, said second disguised matrix Y comprising a second plurality of disguised arguments.

38. (currently amended) The method of claim 37, further comprising the steps of:

outputting said first plurality of disguised arguments and said second plurality of disguised arguments from said first computer for performance of a matrix multiplication computation with said first plurality of disguised arguments and said second plurality of disguised arguments as inputs to said matrix multiplication computation; and

receiving, with said first computer, a result of said matrix multiplication computation.

39. (currently amended) The method of claim 38, further comprising, after the receiving step, the step of:

computing an actual answer from said result, said actual answer comprising said multiplicative product of said first actual matrix M1 and said second actual matrix M2.

40. (currently amended) The method of claim 38, further comprising the steps of:

receiving said first plurality of disguised arguments and said second plurality of disguised arguments at a second computer;

performing said matrix multiplication computation with said second computer; and

sending said result from said second computer to said first computer, said result being in a disguised form relative to an answer that would have been obtained by submitting said first plurality of actual arguments and said second plurality of actual arguments to said second computer.

41. (previously presented) The method of claim 37, wherein said first actual matrix M1 is a non-square matrix.

42. (previously presented) The method of claim 37, wherein said second actual matrix M2 is a non-square matrix.

43. (currently amended) The method of claim 37, further comprising, before the step of ~~computing~~ disguising said first ~~disguised matrix X~~ plurality of actual arguments, the step of:

changing a dimension of said first actual matrix M1.

44. (currently amended) The method of claim 37, further comprising, before the step of ~~computing~~ disguising said second ~~disguised matrix Y~~ plurality of actual arguments, the step of:

changing a dimension of said second actual matrix M2.

45. (currently amended) The method of claim 37, further comprising the steps of:

preparing, in said memory of said first computer, a first ~~random~~ matrix S_1 comprising a first plurality of ~~random~~ pseudorandom arguments, and a second ~~random~~ matrix S_2 comprising a second plurality of ~~random~~ pseudorandom arguments;

generating, with said first computer, a first ~~random~~ pseudorandom number β, a second ~~random~~ pseudorandom number γ, a third ~~random~~ pseudorandom number β', and a fourth ~~random~~ pseudorandom number γ';

computing, with a second computer, matrices W, U, and U' as follows:

$$W = (X + S_1)(Y + S_2),$$

$$U = (\beta X - \gamma S_1)(\beta Y - \gamma S_2), \text{ and}$$

$$U' = (\beta' X - \gamma' S_1)(\beta' Y - \gamma' S_2);$$

computing, with said first computer, matrices V and V' as follows:

$$V = (\beta + \gamma)^{-1} (U + \beta\gamma W), \text{ and}$$

$$V' = (\beta' + \gamma')^{-1} (U' + \beta'\gamma' W);$$

computing, with said second computer, a matrix Z as follows:

$$Z = (\gamma'\beta - \gamma\beta')^{-1} (\gamma' V - \gamma V'); \text{ and}$$

deriving, with said first computer, [[a]] said multiplicative product of said first actual matrix M1 and said second actual matrix M2 from said matrix Z.

46. (previously presented) The method of claim 45, wherein said first actual matrix M1 is a non-square matrix.

47. (previously presented) The method of claim 45, wherein said second actual matrix M2 is a non-square matrix.

48. (currently amended) The method of claim 45, further comprising, before the step of ~~computing~~ disguising said first ~~disguised matrix X~~ plurality of actual arguments, the step of:

changing a dimension of said first actual matrix M1.

49. (currently amended) The method of claim 45, further comprising, before the step of ~~computing~~ disguising said second ~~disguised matrix Y~~ plurality of actual arguments, the step of:

changing a dimension of said second actual matrix M2.

50. (currently amended) A method for outsourcing a matrix inversion computation from a first computer to a second computer, wherein the contents of a matrix are disguised prior to delivery of the matrix to the second computer for the matrix inversion computation thereby hindering discovery of the contents of the matrix by the second computer and unauthorized parties, the method comprising the steps of:

(a) providing, in a memory of a first computer, a first actual matrix M comprising a first plurality of actual arguments, wherein an inverse of said first actual matrix M is desired;

(b) providing, in said memory of said first computer, a ~~random~~ matrix S comprising a plurality of ~~random~~ pseudorandom arguments;

(c) generating, with said first computer, a first disguising matrix P~~1~~,1 and a second disguising matrix P2, ~~a third disguising matrix P3, a fourth disguising matrix P4, and a fifth disguising matrix P5,~~ wherein each said disguising matrix is a sparse matrix comprising at least one non-zero disguising argument, and wherein each said at least one non-zero disguising argument comprises a ~~random~~ pseudorandom number;

(d) disguising said first plurality of actual arguments by computing, with said first computer, a matrix Q as follows:

$$Q = P1 * M * S * P2^{-1}; \text{ ~~and,~~}$$

said matrix Q comprising a first plurality of disguised arguments;

(e) transmitting said matrix Q to a second computer; and

(f) ~~(e)~~ determining, with [[a]] said second computer, whether said matrix Q is invertible.

51. (currently amended) The method of claim 50, wherein said matrix Q is determined to be invertible, the method further comprising the steps of:

(g) ~~(f)~~ inverting, with said second computer, said matrix Q to create an inverse matrix Q^{-1};

(h) generating, with said first computer, a third disguising matrix P3, a fourth disguising matrix P4, and a fifth disguising matrix P5, wherein each said

disguising matrix is a sparse matrix comprising at least one non-zero disguising argument, and wherein each said at least one non-zero disguising argument comprises a pseudorandom number;

(i) ~~(g)~~ computing, with said first computer, a matrix T as follows:

$$T = P4 * P2^{-1} * Q^{-1} * P1 * P5^{-1};$$

(j) ~~(h)~~ computing, with said first computer, a matrix R as follows:

$$R = P3 * S * P4^{-1};$$

(k) ~~(i)~~ computing, with said second computer, a matrix Z as follows:

$$Z = R * T; \text{ and}$$

(l) ~~(j)~~ deriving, with said first computer, ~~an~~ said inverse of said first actual matrix M from said matrix Z.

52. (currently amended) The method of claim 50, wherein said matrix Q is determined not to be invertible, the method further comprising the steps of:

(g) ~~(f)~~ computing, with said first computer, a matrix S' as follows: $S' = S_1*S*S_2$, where S_1 and S_2 are matrices known to be invertible;

(h) ~~(g)~~ determining, with said second computer, whether said matrix S' is invertible; and

(i) ~~(h)~~ if said matrix S' is determined not to be invertible, repeating steps (b)-(f) [[(e)]].

53. (currently amended) The method of claim 50, further comprising, before the step of providing said ~~random~~ matrix S, the step of:

changing a dimension of said first actual matrix M.

54. (currently amended) A method for outsourcing the computation of a solution vector for a system of linear equations of the form Mx=b from a first computer to a second computer, wherein ~~M is~~ the system of linear equations is disguised prior to delivery of the system of linear equations to the second computer for computation of the solution vector, thereby hindering discovery of the system of linear equations by the second computer and unauthorized parties, the method comprising the steps of:

providing a matrix M, said matrix M having *n* rows and *n* columns and comprising a first plurality of actual arguments, wherein *n* is a positive integer~~, wherein b is a vector of dimension *n*, and wherein x is a solution vector, the method comprising the steps of:~~;

providing a vector b, said vector b having *n* elements and comprising a second plurality of actual arguments;

generating, with a first computer, a ~~random~~ matrix B having ~~the same~~ dimensions ~~as~~ identical to said matrix M ~~and~~, said matrix B comprising a plurality of ~~random~~ pseudorandom arguments;

~~generating, with said first computer, a random integer j such that $1 \leq j \leq n$;~~

disguising said second plurality of actual arguments by replacing, in a memory of said first computer, a row of said ~~random~~ matrix B ~~corresponding to said random integer j~~ with said vector b;

preparing, in said memory of said first computer, at least two disguising matrices, each said disguising matrix being a sparse matrix comprising at least one non-zero disguising argument, wherein each said at least one non-zero disguising argument comprises a ~~random~~ pseudorandom number;

disguising said first plurality of actual arguments by computing, with said first computer, a matrix C' using said matrix M and at least two of said at least two disguising matrices, said matrix C' comprising a first plurality of disguised arguments;

disguising said second plurality of actual arguments by computing, with said first computer, a matrix G' using said ~~random~~ matrix B and at least two of said at least two disguising matrices; and

~~computing, with a second computer, a matrix U as follows: $U = C'^{-1} * G'$;~~

~~computing, with said first computer, a matrix X using said matrix U and at least two of said at least two disguising matrices; and~~

~~deriving, with said first computer, said solution vector x from said matrix X.~~

transmitting said matrix C' and said matrix G' to a second computer.

55. (currently amended) A method for outsourcing the computation of a solution vector for a system of linear equations of the form Mx=b from a first computer to a second computer, wherein ~~M is~~ the system of linear equations is disguised prior to delivery of the system of linear equations to the second computer for computation of the solution vector, thereby hindering discovery of the system of linear equations by the second computer and unauthorized parties, the method comprising the steps of:

providing a matrix M, said matrix M having *n* rows and *n* columns and comprising a first plurality of actual arguments, wherein *n* is a positive integer~~, wherein b is a vector of dimension *n*, and wherein x is a solution vector, the method comprising the steps of:~~;

providing a vector b, said vector b having *n* elements and comprising a second plurality of actual arguments;

disguising said first plurality of actual arguments by embedding, in a memory of a first computer, ~~embedding~~ said matrix M in a larger matrix M';

disguising said second plurality of actual arguments by embedding, in said memory of said first computer, ~~embedding~~ said vector b in a larger vector b';

computing a solution vector x' that satisfies the equation M'x' = b', wherein said computation of said solution vector x' is allocated between said first computer and a second computer with each of said first computer and said second computer performing at least a portion of said computation of said solution vector x'; and

deriving, with said first computer, ~~said~~ a solution vector x from said solution vector x', said solution vector x satisfying the equation Mx = b.

56. (currently amended) A method for outsourcing the computation of an estimate for the solution of a quadrature computation of the form

$$\int_a^b f(x)dx,$$

from a first computer to a second computer, wherein the estimate to be computed must conform to a predetermined level of accuracy, and wherein function f(x) is disguised to hinder discovery of function f(x) by the second computer and unauthorized parties, the method comprising the steps of:

creating, in a memory of a first computer, at least seven numbers y_i, wherein i is an integer index variable having a property of $1 \leq i \leq max$, wherein $y_1 = a$ and $y_{max} = b$, and wherein said other numbers y_i satisfy the following properties:

$a < y_i < b$, and

$y_{i-1} < y_i$;

creating, in said memory of said first computer, a number of values v_i, wherein i is an integer index variable having a property of $1 \leq i \leq max$, wherein there are the same number of values v_i as numbers y_i, and wherein said values v_i satisfy the following properties:

$v_{i-1} < v_i$, and

min $|f(x)| \approx v_1 \leq v_{max} \approx \max |f(x)|$, wherein the operations min $|f(x)|$ and max $|f(x)|$ return the minimum and maximum absolute value of function f(x), respectively;

creating, in said memory of said first computer, a cubic spline g(y) with breakpoints comprising said numbers y_i such that $g(y_i) = v_i$;

integrating, with said first computer, cubic spline g(y) from a to b to obtain a value I_1;

creating disguised function h(x), said disguised function h(x) being created using said function f(x) and said cubic spline g(y);

transmitting said disguised function h(x) and a designation of said predetermined level of accuracy to a second computer;

computing, with ~~a~~ said second computer, a value I_2 using said ~~cubic spline g(y),~~ ~~said~~ disguised function [[f(x),]] h(x) and [[a]] said designation of said predetermined level of accuracy; and

computing, with said first computer, said estimate by subtracting said value I_1 from said I_2.

57. (currently amended) A method for outsourcing a convolution computation of two vectors from a first computer to a second computer, wherein the contents of the vectors are disguised prior to delivery of the vectors to the second computer for the convolution computation, thereby

hindering discovery of the contents of the vectors by the second computer and unauthorized parties, the method comprising the steps of:

providing, in a memory of a first computer, a first vector M_1 and a second vector M_2, said first vector M_1 and said second vector M_2 being of equivalent size;

creating, in said memory of said first computer, a first ~~random~~ pseudorandom vector S_1 and a second ~~random~~ pseudorandom vector S_2, said first ~~random~~ pseudorandom vector S_1 and second ~~random~~ pseudorandom vector S_2 being the same size as said first vector M_1 and said second vector M_2;

generating, with said first computer, a first ~~random~~ pseudorandom number α, a second ~~random~~ pseudorandom number β, a third ~~random~~ pseudorandom number γ, a fourth ~~random~~ pseudorandom number β', and a fifth ~~random~~ pseudorandom number γ'; and

~~computing, with a second computer, convolutions W, U, and U' as follows:~~

~~$W = (\alpha M_1 + S_1) \otimes (\alpha M_2 + S_2)$,~~

~~$U = (\beta M_1 + \gamma S_1) \otimes (\beta M_2 + \gamma S_2)$, and~~

~~$U' = (\beta' M_1 + \gamma' S_1) \otimes (\beta' M_2 + \gamma' S_2)$;~~

disguising said first vector M_1 and said second vector M_2 by generating vector D, vector E, vector F, vector G, vector H, and vector I as follows: $D = \alpha M_1$, $E = \alpha M_2$, $F = \beta M_1$, $G = \beta M_2$, $H = \beta' M_1$, and $I = \beta' M_2$.

~~computing, with said first computer, vectors V and V' as follows:~~

~~$V = (\beta + \alpha\gamma)^{-1}(\alpha U + \beta\gamma W)$, and~~

~~$V' = (\beta' + \alpha\gamma')^{-1}(\alpha U' + \beta'\gamma' W)$; and~~

~~deriving, with said first computer, a convolution of vectors M1 and M2 utilizing vectors V and V'.~~

58. (currently amended) The method of claim 57, further comprising, before the step of creating said first ~~random~~ pseudorandom vector S_1 and said second ~~random~~ pseudorandom vector S_2, the step of:

altering, in said memory of said first computer, said size of said first vector M_1 and said second vector M_2.

59. (currently amended) A method for outsourcing the computation of a solution to a linear differential equation $Ly = F(x,y)$ of order N from a first computer to a second computer, wherein N is a positive integer, the linear differential equation having boundary conditions $y(x_i) = y_i$, wherein i is an integer index variable and $0 \leq i < N$, wherein the linear differential equation and boundary conditions are disguised prior to delivery of the linear differential equation and boundary conditions to the second computer for the convolution computation, thereby hindering discovery of the linear differential equation and boundary conditions by the second computer and unauthorized parties, the method comprising the steps of:

creating, in a memory of a first computer, a cubic spline $g(x)$ and a function $u(x) = Lg(x)$, wherein $Lg(x)$ is a linear differential equation of order N;

disguising said linear differential equation Ly by adding said function u(x) thereto;

disguising said boundary conditions $y(x_i)$ by adding function $u(x_i)$ thereto;

transmitting said disguised linear differential equation Ly and said disguised boundary conditions $y(x_i)$ to a second computer;

deriving, with ~~a~~ said second computer, a solution function z(x) ~~for the following set of equations:~~ from said disguised linear differential equation Ly ~~= F(x,y) + u(x),~~ and said disguised boundary conditions $y(x_i)$ ~~= y_i + u~~;

deriving, with said first computer, a solution to said linear differential equation Ly from said ~~derived~~ solution function z(x).

60. (currently amended) A method for disguising a symbolic mathematical expression in a software program, the method comprising the step of:

automatically applying one or more transformation techniques to said symbolic mathematical expression, said one or more transformation techniques selected from the group consisting of disguising constants in said symbolic mathematical expression and replacing variable names in said symbolic mathematical expression.

61. (currently amended) A method for disguising a symbolic mathematical expression in a software program, the method comprising the step of:

automatically applying one or more transformation techniques to said symbolic mathematical expression, said one or more transformation techniques selected from the group consisting of applying at least one mathematical identity function to said symbolic mathematical expression and applying at least one expansion of unity to said symbolic mathematical expression.

62. (currently amended) A method for ~~outsourcing~~ analyzing a digital image wherein a computation for detecting edges of ~~an image, wherein the image is~~ the digital image is outsourced from a first computer to a second computer, and wherein the digital image is disguised prior to delivery of the digital image to the second computer for the edge detection computation thereby hindering discovery of the digital image by the second computer and unauthorized parties, the method comprising the steps of:

providing a digital image, said digital image being represented by an *nxn* array of pixel values p(x,y) between 0 and ~~100,000~~*N* on a unit square $0 \leq x,y \leq 1$, wherein *n* ~~is a~~ and *N* are positive ~~integer, the method comprising the steps of:~~ integers;

creating, in a memory of a first computer, at least ten ordered number pairs x_i,y_i, wherein *i* is an integer index variable having a property of $1 \leq i \leq max$, wherein $x_1,y_1 = 0$,

wherein $x_{max},y_{max}= 1$, and wherein the remaining ordered number pairs x_i,y_i satisfy the following properties:

$0 < x_i,y_i < 1$, and

$x_{i-1},y_{i-1} < x_i,y_i$;

creating, in ~~a~~ said memory of ~~a~~ said first computer, a plurality of ~~random~~ pseudorandom values $v_{i,j}$ such that $0 \leq v_{i,j} \leq$ ~~50,000~~ $N/2$;

creating, in ~~a~~ said memory of [[a]] said first computer, four number pairs a_k,b_k, wherein k is an integer index variable having a property of $1 \leq k \leq 4$, each said number pair a_k,b_k comprising positive ~~random~~ pseudorandom numbers such that $a_1 = \min(a_k)$, $a_4 = \max(a_k)$, $b_1 = \min(b_k)$, and $b_4 = \max(b_k)$;

creating, in ~~a~~ said memory of said first computer, a bi-cubic spline s(x,y), such that each $s(x_i,y_i) = v_{i,j}$;

determining, with said first computer, a linear change of coordinates from (x,y) coordinates to (u,v) coordinates that maps said unit square into a rectangle with vertices (a_k,b_k);

disguising said pixel values p(x,y) by converting, with said first computer, said pixel values p(x,y) to pixel values p(u,v);

disguising said bi-cubic spline s(x,y) by converting, with said first computer, said bi-cubic spline s(x,y) to a bi-cubic spline s(u,v); and

~~computing, with said second computer, an image e(u,v) using~~ transmitting said pixel values p(u,v) and said bi-cubic spline s(u,v)~~; and~~ to a second computer.

~~computing, with said first computer, edges of said image e(u,v).~~

63. (currently amended) A method for ~~outsourcing a template matching computation for image analysis, the template matching computation of a score matrix $C_{I,P}$, the score matrix $C_{I,P}$ comprising~~ analyzing a digital image wherein the desired outcome of the analysis is an approximation of whether ~~an~~ a digital image object ~~P of size *nxn* appears in a larger image I of size *NxN*, wherein *n* and *N* are positive integers, the method comprising the steps of:~~ appears in a larger digital image, wherein a portion of the analysis is outsourced from a first computer to a second computer, and wherein the digital image object and the larger digital image are disguised prior to delivery of the digital image object and the larger digital image to the second computer for the analysis thereby hindering discovery of the image object and the larger image by the second computer and unauthorized parties, the method comprising the steps of:

providing an image I, said image I being represented by a matrix of size *N*x*N*, wherein *N* is a positive integer;

providing an image object P, said image object P being represented by matrix of size *n*x*n*, wherein *n* is a positive integer, and wherein $n < N$;

generating, with a first computer, a ~~random~~ matrix S1 of size *N*x*N*, ~~and a random~~ said matrix S1 comprising a plurality of pseudorandom arguments;

generating, with a first computer, a matrix S2 of size *n*x*n*, said matrix S2 comprising a plurality of pseudorandom arguments;

generating, with said first computer, first ~~random~~ pseudorandom number α, second ~~random~~ pseudorandom number β, third ~~random~~ pseudorandom number γ, fourth ~~random~~ pseudorandom number β', and fifth ~~random~~ pseudorandom number γ';

disguising said image object P and said image I by computing, with said first computer, a set of disguised arguments comprising the following matrices:

$G = \alpha I + S1$,

$H = \alpha P + S2$,

$J = \beta I - \gamma S \gamma S1$,

$K = \beta P - \gamma S \gamma S2$,

$L = \beta' I - \gamma' S1$, and

$M = \beta' P - \gamma' S2$; and

~~computing, with~~ transmitting said matrices G, H, J, K, L, and M to a second computer.

~~, matrices W, U, and U' as follows:~~

~~$W = C_{(\alpha I + S1),(\alpha P + S2)}$;~~

~~$U = C_{(\beta I - \gamma S1)(\beta P - \gamma S2)}$; and~~

~~$U' = C_{(\beta' I - \gamma' S1),(\beta' P \gamma' S2)}$;~~

~~computing, with said first computer, matrices V and V' as follows:~~

~~$V = (+ \alpha\gamma)^{+}(\alpha U + W)$, and~~

~~$V' = (I + \alpha\gamma)^{-1}(\alpha U' + W)$; and deriving, with said first computer, said score matrix $C_{i,p}$ from said matrices V and V'.~~

64. (currently amended) A method for ~~securely~~ outsourcing the sorting of a ~~sequence~~ plurality of numbers from a first computer to a second computer, wherein the plurality of numbers is disguised prior to delivery of the plurality of numbers to the second computer thereby hindering discovery of the plurality of numbers by the second computer and unauthorized parties, the method comprising the steps of:

providing a set of *n* numbers E a plurality of *n* numbers $E = \{e_1,..., e_n\}$, wherein *n* is a positive integer~~, the method comprising the steps of:~~ ;

selecting, with a first computer, a strictly increasing function f();

generating, with said first computer, a ~~random~~ sorted sequence $\{\Lambda = \lambda_1,\ldots,\lambda_n\}$ of *n* pseudorandom numbers;

disguising said plurality of numbers E by computing, with said first computer, ~~sequence~~ a plurality of numbers E' = f(E) and sequence $\Lambda' = f(\Lambda)$, ~~where~~ wherein f(E) is ~~said sequence~~ obtained from plurality of numbers E by replacing every element e_i ~~by~~ of plurality of numbers E with $f(e_i)$;

computing, with said first computer, set W ~~as follows: W = E' U A'~~; by concatenating said plurality of numbers E' and said sequence Λ'; and

disguising set W by randomly permuting set W with said first computer~~;~~.

~~sorting, with a second computer, randomly permuted set W to derive sorted set W'; and~~

~~deriving, with said first computer, sorted sequence E from sorted set W'.~~

65. (currently amended) A method for ~~secure outsourcing of~~ text string analysis wherein it is desired to determine whether a text ~~string pattern matching computation, wherein T is a text string~~ pattern appears in a larger text string, wherein a portion of the analysis is outsourced from a first computer to a second computer, and wherein the text pattern and the text string are disguised prior to delivery of the text pattern and the text string to the second computer thereby hindering discovery of the text pattern and the text string by the second computer and unauthorized parties, the method comprising the steps of:

(a) providing a text string T of length *N*, wherein *N* is a positive integer, said text string T comprising *N* text symbols~~, and wherein P is~~;

(b) providing a text pattern P of length *n*, wherein *n* is a positive integer that is smaller than *N*, said text pattern P comprising *n* text symbols~~, and wherein~~;

(c) providing an alphabet A, said alphabet A ~~comprises the~~ comprising a plurality of possible text symbols that could appear in said text string T or said text pattern P~~, the method comprising the steps of:~~;

(d) ~~(a)~~ selecting a text symbol from said alphabet A;

(e) generating disguised text string T_x by replacing, with a first computer, each instance of said selected text symbol in said text string T with the number 1, and replacing each other text symbol in said text string T with the number 0;

(f) ~~(b)~~ generating disguised text pattern P_x by replacing, with a first computer, each instance of said selected text symbol in said text ~~string T~~ pattern P with the number 1, and replacing each other text symbol in text ~~string T with the number 0, the resultant text string being designated T_x;~~ pattern P with the number 0; and

~~(c) replacing, with a first computer, each instance of said selected text symbol in said text pattern P with the number 1, and replacing each other text symbol in text pattern P with the number 0, the resultant text pattern being designated P_x;~~

(g) ~~(d)~~ augmenting, with a first computer, said text pattern P_x into a longer text string P' of length N by adding zeros thereto;

~~(e) computing, with a second computer, a value D_* as follows:~~

~~$D_*(i) = T_*(i + k)P'(k), \; 0 \leq i \leq N - n$;~~

~~(f) repeating steps (a)-(e) until all text symbols from said alphabet A have been selected one time; and~~

~~(g) computing, with said first computer, a score matrix $C_{T,P}$ using all said values D_*.~~

(h) transmitting said text string T_x and said text string P' to a second computer.

66. (currently amended) A first computer for disguising the contents of a matrix prior to delivery of the matrix to a second computer for a matrix multiplication computation thereby hindering discovery of the matrix by the second computer and unauthorized parties, the computer comprising:

a memory;

computer circuitry configured to define a first actual matrix M1 in said memory, said first actual matrix M1 comprising a first plurality of actual arguments~~, and~~;

computer circuitry configured to define a second actual matrix M2 in said memory, said second actual matrix M2 comprising a second plurality of actual arguments;

computer circuitry configured to prepare at least two disguising matrices in said memory, each said disguising matrix being a sparse matrix comprising at least one non-zero disguising argument, wherein each said at least one non-zero disguising argument comprises a ~~random~~ pseudorandom number;

computer circuitry configured to ~~compute~~ disguise said first plurality of actual arguments by computing a first disguised matrix X using said first actual matrix M1 and at least one said disguising matrix, said first disguised matrix X comprising a first plurality of disguised arguments; and

computer circuitry configured to ~~compute~~ disguise said second plurality of actual arguments by computing a second disguised matrix Y using said second actual matrix M2

and at least one disguising matrix, said second disguised matrix Y comprising a second plurality of disguised arguments.

67. (currently amended) The computer of claim 66, further comprising:

computer circuitry configured to output said first plurality of disguised arguments and said second plurality of disguised arguments for performance of a matrix multiplication~~;~~ computation by another computer, wherein said first plurality of disguised arguments and said second plurality of disguised arguments comprise inputs to said matrix multiplication computation;

computer circuitry configured to receive a result of said matrix multiplication computation from said other computer; and

computer circuitry configured to compute an actual answer from said result, said actual answer comprising said product of said first actual matrix M1 and said second actual matrix M2.

68. (currently amended) The computer of claim 66, further comprising:

computer circuitry configured to prepare a first ~~random~~ pseudorandom matrix S_1 comprising a first plurality of ~~random~~ pseudorandom arguments, and to prepare a second ~~random~~ pseudorandom matrix S_2 comprising a second plurality of ~~random~~ pseudorandom arguments;

computer circuitry configured to generate a first ~~random~~ pseudorandom number β, a second ~~random~~ pseudorandom number γ, a third ~~random~~ pseudorandom number β', and a fourth ~~random~~ pseudorandom number γ';

computer circuitry configured to compute a first set of disguised arguments from the following matrix operations:

$$(X + S_1),$$

$$(Y + S_2),$$

$$(\beta X - \gamma S_1),$$

$$(\beta Y - \gamma S_2),$$

$$(\beta' X - \gamma' S_1), \text{ and}$$

$$(\beta' Y - \gamma' S_2);$$

computer circuitry configured to output said first set of disguised arguments;

computer circuitry configured to receive a matrix U, a matrix U', and a matrix W computed from said first set of disguised arguments;

computer circuitry configured to compute a second set of disguised arguments comprising matrices V and V' as follows:

$$V = (\beta + \gamma)^{-1} (U + \beta\gamma W), \text{ and}$$

$$V' = (\beta' + \gamma')^{-1} (U' + \beta'\gamma' W);$$

computer circuitry configured to output said second set of disguised arguments;

computer circuitry configured to receive a matrix Z computed from said second set of disguised arguments; and

computer circuitry configured to derive a multiplicative product of said first actual matrix M1 and said second actual matrix M2 from said matrix Z.

69. (currently amended) A computer for disguising a matrix prior to transmitting the matrix to another computer for a matrix inversion computation thereby hindering discovery of the matrix by the other computer and unauthorized parties, the computer comprising:

a memory;

computer circuitry configured to define a ~~first actual~~ matrix M in said memory, said matrix M comprising a first plurality of actual arguments;

computer circuitry configured to prepare a ~~random~~ pseudorandom matrix S, said pseudorandom matrix S comprising a plurality of ~~random~~ pseudorandom arguments;

computer circuitry configured to generate a first disguising matrix P~~1~~,1 and a second disguising matrix P2, ~~a third disguising matrix P3, a fourth disguising matrix P4, and a fifth disguising matrix P5,~~ wherein each said disguising matrix is a sparse matrix comprising at least one non-zero disguising argument, wherein each said at least one non-zero disguising argument comprises a ~~random~~ pseudorandom number; and

computer circuitry configured to ~~compute~~ disguise said matrix M by computing a matrix Q as follows:

$$Q = P1 * M * S * P2^{-1}.$$

70. (currently amended) The computer of claim 69, further comprising:

computer circuitry configured to receive an inverse matrix Q^{-1} of said matrix Q;

computer circuitry configured to generate a third disguising matrix P3, a fourth disguising matrix P4, and a fifth disguising matrix P5, wherein each said disguising matrix is a sparse matrix comprising at least one non-zero disguising argument, wherein each said at least one non-zero disguising argument comprises a pseudorandom number;

computer circuitry configured to compute a matrix T as follows:

$$T = P4 * P2^{-1} * Q^{-1} * P1 * P5^{-1};$$

computer circuitry configured to compute, a matrix R as follows:

$$R = P3 * S * P4^{-1};$$

computer circuitry configured to output said matrices T and R;

computer circuitry configured to receive a matrix Z, said matrix Z being computed from said matrices T and R; and

computer circuitry configured to derive an inverse of said matrix M from said matrix Z.

71. (currently amended) The computer of claim 69, further comprising:

computer circuitry configured to compute a matrix S' as follows: $S' = S_1 * S * S_2$, wherein S_1 and S_2 are matrices known to be invertible; and

computer circuitry configured to output said matrix S'.

72. (currently amended) A computer for use in the computation of a solution vector for a system of linear equations of the form Mx=b, ~~wherein M is a matrix having~~ the computer being able to disguise the system of linear equations prior to delivery of the system of linear equations to another computer for computation of the solution vector, thereby hindering discovery of the system of linear equations by the other computer and unauthorized parties, the computer comprising:

a memory;

computer circuitry configured to receive a matrix M and store said matrix M in said memory, said matrix M comprising n rows and n columns wherein n is a positive integer, ~~wherein b is a vector of dimension n, and wherein x is a solution vector, the computer comprising:~~ said matrix M comprising a first plurality of actual arguments;

computer circuitry configured to receive a vector b and store said vector b in said memory, said vector b comprising n elements, said vector b comprising a second plurality of actual arguments;

computer circuitry configured to generate a ~~random~~ matrix B having the same dimensions as matrix M and comprising a plurality of ~~random~~ pseudorandom arguments;

computer circuitry configured to ~~generate a random integer j such that $1 \leq j \leq n$; computer circuitry configured to replace~~ disguise said second plurality of actual arguments by replacing a row of said ~~random~~ matrix B ~~corresponding to said random integer~~ j with said vector b;

computer circuitry configured to prepare at least two disguising matrices, each said disguising matrix being a sparse matrix comprising at least one non-zero disguising argument, wherein each said at least one non-zero disguising argument comprises a ~~random~~ pseudorandom number;

computer circuitry configured to ~~compute~~ disguise said first plurality of actual arguments by computing a matrix C' using said matrix M and at least two of said at least two disguising matrices;

computer circuitry configured to ~~compute~~ disguise said second plurality of actual arguments by computing a matrix G' using said ~~random~~ matrix B and at least two of said at least two disguising matrices;

computer circuitry configured to output said matrices C' and G';

computer circuitry configured to receive a matrix U, said a matrix U computed from said matrix G' and the inverse of said matrix C';

computer circuitry configured to compute a matrix X using said matrix U and at least two of said at least two disguising matrices; and

computer circuitry configured to derive ~~said~~ a solution vector x from said matrix X, said solution vector x satisfying the following: Mx=b.

73. (currently amended) A computer for use in the computation of a solution vector x for a system of linear equations of the form Mx=b, ~~wherein M is a matrix having *n* rows and *n* columns wherein *n* is a positive integer, wherein b is a vector of dimension *n*, and wherein x is a~~

~~solution vector~~ the computer being able to disguise the system of linear equations prior to delivery of the system of linear equations to another computer for computation of the solution vector, thereby hindering discovery of the system of linear equations by the other computer and unauthorized parties, the computer comprising:

a memory;

computer circuitry configured to ~~embed~~ receive a matrix M and store said matrix M in said memory, said matrix M comprising a first plurality of actual arguments;

computer circuitry configured to receive a vector b and store said vector b in said memory, said vector b comprising a second plurality of actual arguments;

computer circuitry configured to disguise said first plurality of actual arguments by embedding said matrix M in a matrix M', said matrix M' being larger than said matrix M;

computer circuitry configured to ~~embed~~ disguise said second plurality of actual arguments by embedding said vector b in a vector b', said vector b' being larger than said vector b;

computer circuitry configured to outsource to another computer at least a portion of the computation of a solution vector x' that satisfies equation M'x' = b'; and

computer circuitry configured to derive ~~said~~ a solution vector x from said solution vector x', said solution vector x satisfying the equation Mx = b.

74. (currently amended) A computer for use in disguising a function f(x) to hinder discovery of function f(x) by other computers or unauthorized parties, wherein the function f(x) is used in the computation of an estimate for the solution of a quadrature computation of the form

$$\int_a^b f(x)dx,$$

and wherein the estimate to be computed must conform to a predetermined level of accuracy, the ~~method~~ computer comprising ~~the steps of~~:

a memory;

computer circuitry configured to receive a function f(x) and store said function f(x) in said memory;

computer circuitry configured to generate at least seven numbers y_i, wherein i is an integer index variable having a property of $1 \leq i \leq max$, wherein $y_1 = a$ and $y_{max} = b$, and wherein said other numbers y_i satisfy the following properties:

$a < y_i < b$, and

$y_{i-1} < y_i$;

computer circuitry configured to generate a quantity of values v_i, wherein i is an integer index variable having a property of $1 \leq i \leq max$, wherein there are the same quantity of said values v_i as said numbers y_i, and wherein said values v_i satisfy the following properties:

$v_{i-1} < v_i$, and

min $|f(x)| \approx v_1 \leq v_{max} \approx \max |f(x)|$, wherein the operations min $|f(x)|$ and max $|f(x)|$ return the minimum and maximum absolute value of function f(x), respectively;

computer circuitry configured to generate a cubic spline g(y) with breakpoints comprising cubic spline said numbers y_i such that $g(y_i) = v_i$;

computer circuitry configured to integrate said cubic spline g(y) from a to b to obtain a value I_1;

computer circuitry configured to ~~output said cubic spline g(y), said function f(x),~~create disguised function h(x), said disguised function h(x) being created using said function f(x) and said cubic spline g(y);

computer circuitry configured to output said disguised function h(x) and a designation of said predetermined level of accuracy;

computer circuitry configured to receive a value I_2 from another computer, wherein said value I_2 is computed using said disguised function h(x) and said designation of said predetermined level of accuracy; and

computer circuitry configured to compute said estimate using said value I_1 and said value I_2.

75. (currently amended) A computer for use in disguising two vectors, wherein said vectors are to be used in a convolution computation ~~of two vectors~~, and wherein the contents of the vectors are disguised prior to delivery of the vectors to another computer for the convolution

computation thereby hindering discovery of the contents of the vectors by the other computer and unauthorized parties, the computer comprising:

a memory;

computer circuitry configured to define a first vector M_1 and a second vector M_2~~,~~2 in said memory, said first vector M_1 and said second vector M_2 being of equivalent size;

computer circuitry configured to define a first ~~random~~ pseudorandom vector S_1 and a second ~~random~~ pseudorandom vector S_2, said first ~~random~~ pseudorandom vector S_1 and second ~~random~~ pseudorandom vector S_2 being the same size as said first vector M_1 and said second vector M_2;

computer circuitry configured to define a first ~~random~~ pseudorandom number α, a second ~~random~~ pseudorandom number β, a third ~~random~~ pseudorandom number γ, a fourth ~~random~~ pseudorandom number β', and a fifth ~~random~~ pseudorandom number γ';

computer circuitry configured to disguise said first vector M_1 and said second vector M_2 by generating vector D, vector E, vector F, vector G, vector H, and vector I as follows: $D = \alpha M_1$, $E = \alpha M_2$, $F = \beta M_1$, $G = \beta M_2$, $H = \beta' M_1$, and $I = \beta' M_2$;

computer circuitry configured to disguise said first pseudorandom vector S_1 and said second pseudorandom vector S_2 by generating vector J, vector K, vector L, and vector M as follows: $J = \gamma S_1$, $K = \gamma S_2$, $L = \gamma' S_1$, and $M = \gamma' S_2$;

computer circuitry configured to supply said vectors S_1, S_2, D, E, F, G, H, I, J, K, L, and M to another computer for the computation of a vector U, a vector U', and a vector W as follows:

$W = (D - S_1) \otimes (E + S_2)$,

$U = (F - J) \otimes (G - K)$, and

$U' = (H - L) \otimes (I - M)$;

~~computer circuitry configured to compute a set of disguised arguments from the following matrix operations:~~

~~$(\alpha M_1 - S_1)$;~~

~~$(\alpha M_2 + S_2)$,~~

~~$(\beta M_1 - \gamma S_1)$,~~

~~$(\beta M_2 - \gamma S_2)$,~~

~~$(\beta' M_1 - \gamma' S_1)$, and~~

~~$(\beta' M_2 - \gamma' S_2)$;~~

~~computer circuitry configured to output said set of disguised arguments;~~

computer circuitry configured to receive ~~a matrix U, a matrix U', and a matrix W computed from said disguised arguments;~~ said vector U, said vector U', and said vector W;

computer circuitry configured to derive a convolution of said ~~vectors~~ first vector M_1 and said second vector M_2 using said ~~matrix~~ vector U, said ~~matrix~~ vector U', said ~~matrix W, said first random number α, said second random number β, said third random number γ, said fourth random number β', and said fifth random number γ'~~ vector W.

76. (currently amended) A computer for use in ~~the computation of a solution to~~ disguising a linear differential equation prior to delivery of the linear differential equation to another computer where a solution to the linear differential equation will be solved, thereby hindering discovery of the linear differential equation by the other computer and unauthorized parties, the computer comprising:

a memory;

a linear differential equation Ly = F(x,y) of order *N* stored in said memory, wherein *N* is a positive integer, the linear differential equation having boundary conditions $y(x_i) = y_i$ stored in said memory, wherein *i* is an integer index variable and $0 \leq i < N$~~, the computer comprising:~~;

computer circuitry configured to define a cubic spline g(x) and a function u(x) = Lg(x), wherein Lg(x) is a linear differential equation of order *N*;

computer circuitry configured to ~~output a set of disguised arguments comprising a set of equations as follows:~~ disguise said linear differential equation Ly by adding said function u(x) thereto;

~~Ly = F(x,y) + u(x), and~~

~~$y(x_i) = y_i + u(x_i)$,~~

~~for computation of a solution function for said set of equations; and~~

computer circuitry configured to disguise said boundary conditions $y(x_i)$ by adding function $u(x_i)$ thereto;

computer circuitry configured to transmit said disguised linear differential equation Ly and said disguised boundary conditions $y(x_i)$ to at least one other computer;

computer circuitry configured to receive a solution function z(x) from said at least one other computer, said solution function z(x) being derived from said disguised linear differential equation Ly and said disguised boundary conditions $y(x_i)$; and

computer circuitry configured to derive a solution to said linear differential equation Ly from said solution function~~.~~ z(x).

77. (currently amended) A computer comprising:

a memory, said memory comprising a software program; and

computer circuitry configured to automatically apply one or more transformation techniques to ~~said~~ a symbolic mathematical expression in said software program, said one or more transformation techniques selected from the group consisting of disguising constants in said symbolic mathematical expression and replacing variable names in said symbolic mathematical expression.

78. (currently amended) A computer comprising:

a memory, said memory comprising a software program; and

computer circuitry configured to automatically apply one or more transformation techniques to ~~said~~ a symbolic mathematical expression in said software program, said one or more transformation techniques selected from the group consisting of applying at least

one mathematical identity function to said symbolic mathematical expression and applying at least one expansion of unity to said symbolic mathematical expression.

79. (currently amended) A first computer for use in analyzing a digital image wherein a computation for detecting edges of ~~an image, wherein~~ the digital image is to be outsourced to a second computer, and wherein the digital image is disguised prior to delivery of the digital image to the second computer for the edge detection computation thereby hindering discovery of the digital image by the second computer and unauthorized parties, the first computer comprising:

a memory, said memory comprising a digital image, wherein said digital image is represented in said memory by an nxn array of pixel values p(x,y) between 0 and ~~100,000~~N on a unit square $0 \leq x,y \leq 1$, wherein n ~~is a~~ and N are positive ~~integer, the computer comprising:~~ integers;

computer circuitry configured to define at least ten ordered number pairs x_i,y_i in said memory, wherein i is an integer index variable having a property of $1 \leq i \leq max$, wherein $x_1,y_1 = 0$, wherein $x_{max},y_{max} = 1$, and wherein the remaining ordered number pairs x_i,y_i satisfy the following properties:

$0 < x_i,y_i < 1$, and

$x_{i-1},y_{i-1} < x_i,y_i$;

computer circuitry configured to define a plurality of ~~random~~ pseudorandom values $v_{i,j}$ such that $0 \leq v_{i,j} \leq$ ~~50,000;~~ $N/2$ in said memory;

computer circuitry configured to define four number pairs a_k,b_k in said memory, wherein k is an integer index variable having a property of $1 \leq k \leq 4$, each said number pair a_k,b_k comprising positive ~~random~~ pseudorandom numbers such that $a_1 = \min(a_k)$, $a_4 = \max(a_k)$, $b_1 = \min(b_k)$, and $b_4 = \max(b_k)$;

computer circuitry configured to define a bi-cubic spline s(x,y) in said memory, such that $s(x_i,y_i) = v_{i,j}$;

computer circuitry configured to determine a linear change of coordinates from (x,y) coordinates to (u,v) coordinates that maps said unit square into a rectangle with vertices (a_k,b_k);

computer circuitry configured to disguise said pixel values p(x,y) by converting said pixel values p(x,y) to pixel values p(u,v);

computer circuitry configured to disguise said bi-cubic spline s(x,y) by converting said bi-cubic spline s(x,y) to a bi-cubic spline s(u,v);

computer circuitry configured to output said pixel values p(u,v) and said bi-cubic spline s(u,v)~~;~~ to another computer;

computer circuitry configured to receive an image e(u,v) computed using said pixel values p(u,v) and said bi-cubic spline s(u,v); and

computer circuitry configured to derive edges of said image e(u,v).

80. (currently amended) A first computer for use in ~~a template matching computation for image analysis, the template matching comprising computation of a score matrix $C_{I,P}$, the score~~

~~matrix $C_{I,P}$ comprising~~ analyzing a digital image wherein the desired outcome of the analysis is an approximation of whether ~~an~~ a digital image object ~~P of size *nxn* appears in a larger image I of size *NxN*, wherein *n* and *N* are positive integers, the computer comprising:~~ appears in a larger digital image, wherein a portion of the analysis is outsourced from the first computer to a second computer, and wherein the image object and the larger image are disguised prior to delivery of the image object and the larger image to the second computer for the analysis thereby hindering discovery of the image object and the larger image by the second computer and unauthorized parties, the first computer comprising:

a memory, said memory comprising an image I, said image I being represented in said memory by a matrix of size *N*x*N*, wherein *N* is a positive integer

an image object P in said memory, said image object P being represented by matrix of size *n*x*n*, wherein *n* is a positive integer, and wherein $n < N$;

computer circuitry configured to define a ~~random~~ matrix S1 of size *N*x*N*, ~~and~~ said matrix S1 comprising a plurality of pseudorandom arguments;

computer circuitry configured to define a ~~random~~ matrix S2 of size *n*x*n*, said matrix S2 comprising a plurality of pseudorandom arguments;

computer circuitry configured to define, in said memory, a first ~~random~~ pseudorandom number α, a second ~~random~~ pseudorandom number β, a third ~~random~~ pseudorandom number γ, a fourth ~~random~~ pseudorandom number β', and a fifth ~~random~~ pseudorandom number γ';

computer circuitry configured to ~~compute~~ disguise said image I and said image object P by computing a set of disguised arguments comprising the following matrices:

$$G = \alpha I + S1,$$

$$H = \alpha P + S2,$$

$$J = \beta I - [[yS1]]\ \gamma S1,$$

$$K = \beta P - [[yS1]]\ \gamma S2,$$

$$L = \beta' I - \gamma' S1, \text{ and}$$

$$M = \beta' P - \gamma' S2;$$

computer circuitry configured to output said set of disguised arguments~~;~~ to a second computer for computation of a first score matrix, a second score matrix, and a third score matrix, said first score matrix being computed using said matrices G and H, said second score matrix being computed using said matrices J and K, and said third score matrix being computed using said matrices L and M;

computer circuitry configured to receive ~~a~~ said first score matrix ~~U, a~~, said second score matrix ~~U'~~, and ~~a~~ said third score matrix ~~W computed from said set of disguised arguments~~; and

computer circuitry configured to derive ~~said~~ a final score matrix ~~$C_{I,P}$~~ using said first score matrix, said ~~matrix U, said matrix U', said matrix W, said first random number α, said second random number β, said third random number γ, said fourth random number β', and said fifth random number γ'~~ second score matrix, and said third score matrix, said

final score matrix comprising an approximation of whether said image object P appears in said image I.

81. (currently amended) A first computer for ~~securely outsourcing the sorting of a sequence of *n* numbers E = {e_1,..., e_n}~~use in preparing a set of numbers to be outsourced from the first computer to a second computer where the set of numbers will be sorted into a sequence by the second computer, and wherein the set of numbers is disguised prior to delivery of the set of numbers to the second computer thereby hindering discovery of the set of numbers by the second computer and unauthorized parties, the first computer comprising:

a memory, said memory comprising a set of *n* numbers E, wherein *n* is a positive integer~~, the computer comprising:~~;

computer circuitry configured to define a strictly increasing function f()~~;~~ in said memory;

computer circuitry configured to define, in said memory, a ~~random~~ sorted sequence $\{\Lambda = \lambda_1,\ldots,\lambda_n\}$ of *n* pseudorandom numbers;

computer circuitry configured to ~~compute sequence~~ disguise said set of numbers E by computing a set of *n* numbers E' = f(E) and a sequence *n* numbers Λ' = f(Λ), wherein f(E) is a ~~sequence~~ set of numbers obtained from E by replacing every element e_i by f(e_i), and f(Λ) is a sequence of numbers obtained from Λ by replacing every element λ_i by f(λ_i);

computer circuitry configured to compute ~~set W as follows: W = E' U Λ';~~ a set of numbers W by concatenating said plurality of numbers E' and said sequence Λ';

computer circuitry configured to randomly permute set W ~~with said first computer;~~;

computer circuitry configured to output said randomly permuted set W~~;~~ for sorting by a second computer;

computer circuitry configured to receive sorted set W' from said second computer, said sorted set W' being derived ~~from~~ by sorting said randomly permuted set W; and

computer circuitry configured to derive sorted sequence E'' = $\{e_1,..., e_n\}$ from said sorted set W'.

82. (currently amended) A ~~computer for use in~~ first computer for use in text string analysis wherein it is desired to determine whether a text pattern appears in a larger text string, and wherein the text pattern and the text string are disguised prior to delivery of the text pattern and the text string to a second computer for a text string pattern matching computation, ~~wherein~~ thereby hindering discovery of the text pattern and the text string by the second computer and unauthorized parties, the first computer comprising:

a memory, said memory comprising a test string T is a text string of length *N*, wherein *N* is a positive integer, said text string T comprising *N* text symbols~~, and wherein P is~~;

a text pattern P in said memory, said text pattern P being of length *n*, wherein *n* is a positive integer that is smaller than *N*, said text pattern P comprising *n* text symbols~~, and wherein~~;

an alphabet A in said memory, said alphabet A ~~comprises~~ comprising the plurality of possible text symbols that could appear in text string T or text pattern P~~, the computer comprising:~~;

computer circuitry configured to ~~select a~~ iteratively select one text symbol from said alphabet A~~;~~ until all text symbols from said alphabet A have been selected one time;

~~computer circuitry configured to replace each instance of said selected text symbol in said text string T with the number 1, and replacing each other text symbol in text string T with the number 0, the resultant text string being designated T_x;~~

~~computer circuitry configured to replace~~ computer circuitry configured to, for each selected text symbol in said alphabet A:

(i) generate, in said memory, a disguised text string T_x by replacing each instance of said selected text symbol in said text ~~pattern P~~ string T with the number 1, and replacing each other text symbol in text ~~pattern P with the number 0, the resultant text pattern being designated P_x;~~ string T with the number 0,

(ii) generate, in said memory, disguised text pattern P_x by replacing each instance of said selected text symbol in said text pattern P with the number 1, and replacing each other text symbol in said text pattern P with the number 0,

~~computer circuitry configured to~~(iii) augment said text pattern P_x into a text string P' of length N by adding zeros thereto~~;~~,

~~computer circuitry configured to~~(iv) output said text string P' and said text string T_x~~;~~ to a second computer ~~circuitry configured to~~, and

(v) receive a value D_x computed using said text string P' and said text string T_x ~~as follows: $D_x(i) = T_x(i+k)P'(k), \quad 0 \leq i \leq N-n$~~; and

computer circuitry configured to compute a score matrix ~~$C_{T,P}$~~ using said values D_x. for all said text symbols in said alphabet A.

83. (new) The method of claim 54, further comprising the steps of:

computing, with said second computer, a matrix U as follows: $U = C'^{-1} * G'$;

computing, with said first computer, a matrix X using said matrix U and at least two of said at least two disguising matrices; and

deriving, with said first computer, a solution vector x from said matrix X, said solution vector x satisfying the following: Mx=b.

84. (new) The method of claim 57, further comprising the steps of:

computing, with a second computer, convolutions W, U, and U' as follows:

$W = (D - S_1) \otimes (E + S_2)$,

$U = (F - \gamma S_1) \otimes (G - \gamma S_2)$, and

$U' = (H - \gamma' S_1) \otimes (I - \gamma' S_2)$;

computing, with said first computer, vectors V and V' as follows:

$V = (\beta + \alpha\gamma)^{-1} (\alpha U + \beta\gamma W)$, and

$V' = (\beta' + \alpha\gamma')^{-1} (\alpha U' + \beta'\gamma' W)$; and

deriving, with said first computer, a convolution of said first vector M_1 and said second vector M_2 utilizing said vectors V and V'.

85. (new) The method of claim 62, further comprising the steps of:

computing, with said second computer, an image e(u,v) using said pixel values p(u,v) and said bi-cubic spline s(u,v); and

computing, with said first computer, edges of said image e(u,v).

86. (new) The method of claim 63, further comprising the steps of:

computing, with said second computer, matrices W, U, and U' as follows:

$$W = C_{G,H} = \sum_{k=0}^{n-1} \sum_{k'=0}^{n-1} f(G(i + k, j + k'), H(k, k')), \; 0 \le i, j \le N - n,$$

$$U = C_{J,K} = \sum_{k=0}^{n-1} \sum_{k'=0}^{n-1} f(J(i + k, j + k'), K(k, k')), \; 0 \le i, j \le N - n, \text{ and}$$

$$U' = C_{L,M} = \sum_{k=0}^{n-1} \sum_{k'=0}^{n-1} f(L(i + k, j + k'), M(k, k')), \; 0 \le i, j \le N - n;$$

computing, with said first computer, matrices V and V' as follows:

$$V = (\beta + \alpha\gamma)^{-1} (\alpha U + \beta\gamma W), \text{ and}$$

$$V' = (\beta + \alpha\gamma)^{-1} (\alpha U' + \beta' \gamma' W); \text{ and}$$

deriving, with said first computer, said score matrix from said matrices V and V'.

87. (new) The method of claim 63, further comprising the steps of:

computing, with said second computer, a first score matrix using said matrices G and H;

computing, with said second computer, a second score matrix using said matrices J and K;

computing, with said second computer, a third score matrix using said matrices L and M;

receiving said first score matrix, said second score matrix, and said third score matrix at said first computer; and

deriving, with said first computer, a final score matrix using said first score matrix, said second score matrix, and said third score matrix, said final score matrix comprising an approximation of whether said image object P appears in said image I.

88. (new) The method of claim 64, further comprising the steps of:

sorting, with a second computer, said randomly permuted set W to derive sorted set W'; and

deriving, with said first computer, sorted sequence E from sorted set W'.

89. (new) The method of claim 65, further comprising the steps of:

(i) computing, with a second computer, a value D_x as follows:

$$D_x(i) = \sum_{k=0}^{n-1} T_x(i+k)P'(k), \qquad 0 \le i \le N-n;$$

(j) repeating steps (d)-(i) until all text symbols from said alphabet A have been selected one time; and

(k) computing, with said first computer, a score matrix using all said values D_x, said score matrix comprising an approximation of whether said text pattern P appears in said text string T.